

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

READ INSTRUCTIONS
BEFORE COMPLETING FORM

REPORT DOCUMENTATION PAGE		3. RECIPIENT'S CATALOG NUMBER
1. REPORT NUMBER AFIT/CI/NR 88-46	2. GOVT ACCESSION NO.	
4. TITLE (and Subtitle) CONTROLLED DEGRADATION OF RESOLUTION OF HIGH-QUALITY FLIGHT SIMULATION IMAGES FOR TRAINING EFFECTIVENESS EVALUATION		5. TYPE OF REPORT & PERIOD COVERED MS THESIS
7. AUTHOR(s) DENNIS DAVID KAIP		6. PERFORMING ORG. REPORT NUMBER
9. PERFORMING ORGANIZATION NAME AND ADDRESS AFIT STUDENT AT: UNIVERSITY OF MICHIGAN		8. CONTRACT OR GRANT NUMBER(s)
10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS		
11. CONTROLLING OFFICE NAME AND ADDRESS		12. REPORT DATE 1988
13. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office) AFIT/NR Wright-Patterson AFB OH 45433-6583		14. NUMBER OF PAGES 61
15. SECURITY CLASS. (of this report) UNCLASSIFIED		16. DECLASSIFICATION/DOWNGRADING SCHEDULE
17. DISTRIBUTION STATEMENT (of this Report) DISTRIBUTED UNLIMITED: APPROVED FOR PUBLIC RELEASE		
18. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report) SAME AS REPORT		
19. SUPPLEMENTARY NOTES Approved for Public Release: IAW AFR 190-1 LYNN E. WOLAVER Dean for Research and Professional Development Air Force Institute of Technology Wright-Patterson AFB OH 45433-6583 18 July 88		
20. KEY WORDS (Continue on reverse side if necessary and identify by block number)		
21. ABSTRACT (Continue on reverse side if necessary and identify by block number) ATTACHED		

AD-A196 189

DTIC
ELECTE
AUG 04 1988
S D

DD FORM 1 JAN 73 1473

EDITION OF 1 NOV 65 IS OBSOLETE

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

ABSTRACT

The Operations Training Division of the Air Force Human Resources Laboratory is conducting behavioral research studies on the requirements for training effectiveness in flight simulators. One set of these studies will lead to specifications for simulator display resolution in different types of flight training tasks. A method is required to predictably degrade the resolution of computer-generated flight simulator images.

In this paper, the concept of resolution degradation is explored. Three methods for performing resolution degradation are presented: area-weighted averaging, two-dimensional spatial transform, and viewporting with transform. All three permit degrading image resolution to any desired degree.



Accession For	
NTIS	CRA&I <input checked="" type="checkbox"/>
DTIC	TAB <input type="checkbox"/>
Unannounced <input type="checkbox"/>	
Justification	
By	
Distribution	
Availability Codes	
Dist	Availability Codes
A-1	

CONTROLLED DEGRADATION OF RESOLUTION
OF HIGH-QUALITY FLIGHT SIMULATOR IMAGES
FOR TRAINING EFFECTIVENESS EVALUATION

by

Dennis David Kaip

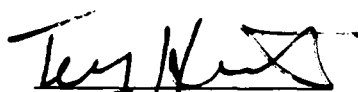
Bachelor of Science in Engineering

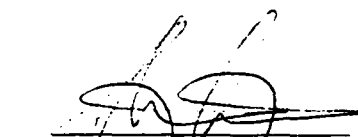
University of Michigan, 1981

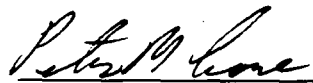
Submitted in Partial Fulfillment of the
Requirements for the Degree of Master of Science
in the Department of Computer Science

University of South Carolina

1988


Department of
Computer Science
Director of Thesis


Department of
Computer Science
2nd Reader


3rd Reader

Dean of the
Graduate School

TABLE OF CONTENTS

ABSTRACT	ii
I. INTRODUCTION	1
II. BACKGROUND	2
III. MODEL OF THE RESOLUTION DEGRADATION	3
IV. MATHEMATICAL FORMULATION OF THE TRANSFORM	9
V. APPLICATION OF A SPATIAL TRANSFORM TECHNIQUE	19
VI. COMPARISON OF TECHNIQUES	28
VII. AN ALTERNATIVE METHOD	45
VIII. SUMMARY	50
LIST OF REFERENCES	52
APPENDIX A	53
APPENDIX B	58

ABSTRACT

The Operations Training Division of the Air Force Human Resources Laboratory (AFHRL) is conducting behavioral research studies on the requirements for training effectiveness in flight simulators. One set of these studies will lead to specifications for simulator display resolution in different types of flight training tasks. A method is required to predictably degrade the resolution of computer-generated flight simulator images.

In this paper, the concept of resolution degradation is explored. Three methods for performing resolution degradation are presented: area-weighted averaging, two-dimensional spatial transform, and viewporting with transform. All three permit degrading image resolution to any desired degree.

I. INTRODUCTION

A flight simulator is a device used to train pilots and air crews without the use of an actual aircraft. The use of flight simulators for training is widespread in both the military and civilian sector. The use of flight simulators has significant advantages over the operation of the actual aircraft. The cost of operating a flight simulator is typically 1/20 the cost of flying the actual aircraft. In addition, pilots and air crews can practice complex and/or dangerous maneuvers in a flight simulator without risking loss of life or aircraft.

As described by Schachter [6], sophisticated flight simulators recreate most of the aspects of flying: aircraft instruments, motion of the aircraft, gravitational forces, radar, and out-the-window views. Most modern flight simulators use computer generated imagery (CGI) to produce the out-the-window views in real-time in response to inputs from flight controls. The imagery can be displayed in a variety of ways: on large CRTs, projected onto the inside of a dome, or through helmet-mounted displays.

Regardless of display type, three things are important to the presentation of CGI: image brightness, contrast, and resolution. Brightness is the luminance of the display.¹ Contrast is the difference in luminance between displayed objects such as an aircraft and the background sky. Resolution is the number of pixels per unit area which make up the display. Considered simplistically, perceptible improvements in any or all of these will result in higher quality, more realistic flight simulator images. Improvements in these are generally limited by cost, technology, and computational efficiency.

Different training tasks are presumed to have different requirements with respect to brightness, contrast, and resolution. The goal for visual display flight training simulators in general, is not realism, but training effectiveness. Therefore, the design requirements for flight simulators should take into consideration the desired training task. (S1)

¹In psychophysics, brightness is the perceived correlate of luminance. In this paper, brightness will be considered equivalent to luminance.

II. BACKGROUND

The Operations Training Division of the Air Force Human Resources Laboratory (AFHRL/OT), at Williams AFB, Arizona, uses sophisticated flight simulators to conduct Research and Development (R&D) on training effectiveness. These flight simulators use CGI to produce high-quality images for out-the-window views. Such flight simulators are typically large, non-transportable, and very expensive.

The Technology Development Branch of AFHRL/OT is conducting behavioral studies to determine requirements and specifications for future flight simulators. The requirements and specifications to be defined include brightness, contrast, and resolution for various flight tasks.

One flight task of interest is the job of intercepting another aircraft. Three of the tasks involved in visual intercept are: detecting the aircraft, determining its aspect orientation (which way it is traveling relative to the viewer), and identifying the aircraft (i.e. F-15, MiG-29, etc.). Accomplishing these tasks occurs at decreasing range (or distance between the viewer and the target aircraft). The AFHRL studies will attempt to define minimum requirements for brightness, contrast, and resolution for these tasks.

In the studies, the brightness and "apparent resolution" will be held constant while the required contrast for several ranges is determined. Various degrees of degraded resolution will be attempted in this process. The term "apparent resolution" is used because in actuality, the resolution of the display device remains the same. For instance, in the research for this paper the display device used was a CRT with a resolution of 768 rows with 1024 pixels per row. The objective is to make the image appear as though it is made up of fewer pixels. Thus the image looks like it is displayed on a lower resolution device. This paper presents a study and evaluation of techniques that can be used to degrade the apparent resolution of an image.

III. MODEL OF THE RESOLUTION DEGRADATION

In order to develop techniques to degrade image resolution, it is important to understand what constitutes a degraded image within the context of this effort. The objective of this work is somewhat contrary to the typical image processing enhancement/restoration goals. As stated by Gonzalez and Wintz [3],

...the ultimate goal of restoration techniques is to improve a given image in some sense.

Restoration techniques typically take an image which has been degraded in some known fashion (e.g. motion blur) and attempt to counter the degradation. This involves developing a model of the process that produced the degradation.

In the case of resolution degradation, techniques do not exist for effectively restoring an image of low resolution. There are enhancement techniques such as edge enhancement and smoothing which may improve the appearance of a low resolution image but cannot restore the higher resolution information. *The low resolution image does not contain the information necessary to recover the high resolution image; the high frequency components have been lost in the sampling to obtain the lower resolution image.* The Nyquist criteria has been violated thus making it impossible to completely recover the original image [4]. If techniques existed for restoring the original appearance of images degraded in resolution, it would be a simple matter to apply the degradation model to a flight simulator image to obtain a degraded image.

As mentioned earlier, the objective of this work is to take a flight simulator image of a given pixel resolution (e.g. 768 x 1024) and display it at an apparent lower resolution (400 x 650). It is not sufficient to simply shrink the picture to fit the desired dimensions. In flight simulators size translates directly to range, so to shrink the image of a target aircraft would simply put that aircraft at greater range. This means that the shrunken image must somehow be translated back to the original screen size with the artifacts of the degradation reasonably intact.

A two phased approach to this problem is appropriate. The first phase will take the original image and shrink it to fit the desired dimensions or pixel resolution. The second process will take the results of the first phase and transform it back to the size of the original image.

Phase One. In the previous discussion of restoration techniques, the idea of resampling was mentioned in conjunction with resolution degradation. When digitizing an image, a discrete sampling of a continuous image is performed. The resolution of the resulting digital image is related to the number of samples taken for a given area of the image. This is often dependent upon the digitizing hardware. It is reasonable to say that the digitizing process degrades the resolution of the original image because the continuous image can be thought of being made up of an infinite number of discrete samples.

In a similar fashion, a digital image can be resampled to produce an image of lower resolution (i.e. fewer samples). To do this, the original image is overlaid with a grid of the desired output size. The value of each grid location is the average of the pixels under it in the original image. A simple example will help illustrate this.

Example 1a.

A very simple 4 x 4 pixel image will be resampled to a 3 x 3. Figure 1 shows the original 4 x 4 image. In order to simplify the averaging, each pixel in the original 4 x 4 image is divided into nine sub-pixels resulting in a 12 x 12 sub-pixel image.

10	10	10	10
10	100	100	10
10	100	100	10
10	10	10	10

Figure 1. Simple 4 x 4 pixel image.

Figure 2 shows the sub-pixel image with the whole pixel borders. As mentioned earlier, the original image is overlaid with a grid of the desired output size. In this case, the 4×4 is overlaid with a 3×3 grid. Figure 3 is the sub-pixel image of Figure 2 overlaid with a 3×3 grid. This results in each pixel of the 3×3 containing twelve sub-pixels. To obtain the value of each pixel in the 3×3 image, its twelve sub-pixels are summed. The result is divided by twelve and rounded to the nearest integer value.

Phase Two. This phase takes the image of smaller size and maps it back to the size of the original image. This phase is quite simply, a reverse of phase one. One might think that a simple reverse of phase one would result the original undegraded image. However, as was mentioned earlier the

Figure 2. Sub-pixel image.

[illegible]

Figure 3. Sub-pixel image with 3 x 3 overlay.

16	33	16
33	100	33
16	33	16

Figure 4. Resulting 3 x 3 pixel image.

resampling process destroys information making it impossible to recover the original image. Thus, in reversing the first phase the image returns to its original size but contains the artifacts of the degradation. To demonstrate this phase the example of phase one is continued.

Example 1b.

Figure 4 shows the 3 x 3 result of phase one. As in phase one, each pixel is divided into sub-pixels. Each pixel in the 3 x 3 is divided into 12 sub-pixels resulting in a 12 x 12 sub-pixel image. Figure 5 shows the resulting sub-pixel image. The 3 x 3 is overlaid with a 4 x 4 grid which is shown in Figure 6. The value of each pixel in the 4 x 4 is the average of the 9 sub-pixels under it. The resulting 4 x 4 image is shown in Figure 7.

16	16	16	16	33	33	33	33	16	16	16	16
16	16	16	16	33	33	33	33	16	16	16	16
16	16	16	16	33	33	33	33	16	16	16	16
16	16	16	16	33	33	33	33	16	16	16	16
33	33	33	33	100	100	100	100	33	33	33	33
33	33	33	33	100	100	100	100	33	33	33	33
33	33	33	33	100	100	100	100	33	33	33	33
33	33	33	33	100	100	100	100	33	33	33	33
16	16	16	16	33	33	33	33	16	16	16	16
16	16	16	16	33	33	33	33	16	16	16	16
16	16	16	16	33	33	33	33	16	16	16	16
16	16	16	16	33	33	33	33	16	16	16	16

Figure 5. 3 x 3 Sub-pixel image.

16	16	16	16	33	33	33	33	16	16	16	16
16	16	16	16	16	33	33	33	33	16	16	16
16	16	16	16	16	33	33	33	33	16	16	16
16	16	16	16	16	33	33	33	33	16	16	16
33	33	33	33	33	100	100	100	100	33	33	33
33	33	33	33	33	100	100	100	100	33	33	33
33	33	33	33	33	100	100	100	100	33	33	33
33	33	33	33	33	100	100	100	100	33	33	33
16	16	16	16	16	33	33	33	33	16	16	16
16	16	16	16	16	33	33	33	33	16	16	16
16	16	16	16	16	33	33	33	33	16	16	16
16	16	16	16	16	33	33	33	33	16	16	16

Figure 6. Sub-pixel image with 4 x 4 overlay.

16	27	27	16
27	61	61	27
27	61	61	27
16	27	27	16

Figure 7. Resulting 4 x 4 pixel image.

IV. MATHEMATICAL FORMULATION OF THE TRANSFORM

The method described by the example of the previous section can be extended to take an image of any arbitrary size and degrade it to any arbitrary degree. To do so involves determining the correct number of sub-pixels which will compose the input and output images. The number of rows of sub-pixels is the least common denominator of the number of rows of pixels in the input image and the number of rows of pixels in the output image. The number of columns of sub-pixels is found in the same fashion. This method works well for very small images such as the example in the previous section, however, it can prove quite cumbersome for larger images. The expense of computing averages of large numbers of sub-pixels can be prohibitive.

The same results can be achieved with an area-weighted averaging technique. In this technique, the intensity (gray-level value) contributed by a pixel in the input image to a pixel in the output image is weighted by the percentage of area of the input pixel contained in the output pixel. This technique can be described by a mathematical formulation as follows:

In this method, the input image f_{in} contains I' rows and J' columns. The value of a pixel in the input picture is represented by $f_{in}(i', j')$. The output image f_{out} contains I rows and J columns. The value of a pixel in the output image is represented by $f_{out}(i, j)$. Given an input image, each pixel of the output image can be described by the following equation:

$$f_{out}(i, j) = \sum_{i'=a}^b \sum_{j'=c}^d w_x(i, i') \cdot w_y(j, j') \cdot f_{in}(i', j')$$

for $0 \leq i \leq I-1$,

$0 \leq j \leq J-1$,

$0 \leq i' \leq I'-1$, and

$0 \leq j' \leq J'-1$

$$\text{where } a = \left\lfloor \frac{I'}{I} \cdot i \right\rfloor,$$

$$c = \left\lfloor \frac{J'}{J} \cdot j \right\rfloor,$$

$\lfloor \rfloor$ describes the floor function,

$$b = \left\lceil \frac{I'}{I} \cdot (i + 1) \right\rceil - 1,$$

$$d = \left\lceil \frac{J'}{J} \cdot (j + 1) \right\rceil - 1,$$

$\lceil \rceil$ describes the ceiling function,

$$w_x = \left| \max\left[i, \frac{I}{I'} \cdot i'\right] - \min\left[i + 1, \frac{I}{I'} \cdot (i' + 1)\right] \right|,$$

and

$$w_y = \left| \max\left[j, \frac{J}{J'} \cdot j'\right] - \min\left[j + 1, \frac{J}{J'} \cdot (j' + 1)\right] \right|$$

In the above equation, each pixel in the output image is determined by summing the contributions of each input pixel influencing it. The limits a, b, c, and d define a rectangular region of pixels in the input image which contribute to the value of the output pixel $f_{out}(i,j)$. The product of $w_x(i,i')$ and $w_y(j,j')$ is the area-weight applied to the input pixel $f_{in}(i',j')$.

Example 2

In this example, the equation presented above will be applied to the image presented in example 1. The image is repeated here as Figure 8. As in example 1, the 4 x 4 image will be transformed to a 3 x 3. The calculation of one of the values in the 3 x 3 will be presented. For this and subsequent examples, the origin ($i = 0$,

$j = 0$) is in the upper left corner. We will find the value of $f_{out}(0,1)$.

The following values are immediately known:

$$I = 3, \quad J = 3,$$

$$I' = 4, \quad J' = 4,$$

$$i = 0, \quad j = 1,$$

$$\frac{I'}{I} = 1.33,$$

and

$$\frac{J'}{J} = 1.33$$

Solving for the limits of the summations we have

$$a = \lfloor 1.33 \cdot 0 \rfloor$$

$$= 0$$

10	10	10	10
10	100	100	10
10	100	100	10
10	10	10	10

Figure 8. Simple 4 x 4 pixel image.

$$\begin{aligned}
b &= \lceil 1.33 \cdot (0 + 1) \rceil - 1 \\
&= \lceil 1.33 \cdot 1 \rceil - 1 \\
&= \lceil 1.33 \rceil - 1 \\
&= 2 - 1 \\
&= 1
\end{aligned}$$

$$\begin{aligned}
c &= \lfloor 1.33 \cdot 1 \rfloor \\
&= \lfloor 1.33 \rfloor \\
&= 1
\end{aligned}$$

$$\begin{aligned}
d &= \lceil 1.33 \cdot (1 + 1) \rceil - 1 \\
&= \lceil 1.33 \cdot 2 \rceil - 1 \\
&= \lceil 2.66 \rceil - 1 \\
&= 3 - 1 \\
&= 2
\end{aligned}$$

Therefore the equation is:

$$f_{\text{out}}(0,1) = \sum_{i'=0}^1 \sum_{j'=1}^2 w_x(0,i') \cdot w_y(1,j') \cdot f_{\text{in}}(i',j')$$

Next we will solve for each value of w_x and each value of w_y .

$$\frac{I}{I'} = 0.75 \quad \text{and} \quad \frac{J}{J'} = 0.75$$

for $i' = 0$,

$$\begin{aligned}
 w_x(0,0) &= \left| \max[0, 0.75 \cdot 0] - \min[1, 0.75 \cdot (0 + 1)] \right| \\
 &= \left| \max[0, 0] - \min[1, 0.75] \right| \\
 &= \left| 0 - 0.75 \right| \\
 &= \left| -0.75 \right| \\
 &= 0.75
 \end{aligned}$$

for $i' = 1$,

$$\begin{aligned}
 w_x(0,1) &= \left| \max[0, 0.75 \cdot 1] - \min[1, 0.75 \cdot (1 + 1)] \right| \\
 &= \left| \max[0, 0.75] - \min[1, 1.5] \right| \\
 &= \left| 0.75 - 1 \right| \\
 &= \left| -0.25 \right| \\
 &= 0.25
 \end{aligned}$$

for $j' = 1$,

$$\begin{aligned}
 w_y(1,1) &= \left| \max[1, 0.75 \cdot 1] - \min[2, 0.75 \cdot (1 + 1)] \right| \\
 &= \left| \max[1, 0.75] - \min[2, 1.5] \right| \\
 &= \left| 1 - 1.5 \right| \\
 &= \left| -0.5 \right| \\
 &= 0.5
 \end{aligned}$$

for $j' = 2$,

$$\begin{aligned}w_y(1,2) &= \left\lfloor \max[1, 0.75 \cdot 2] - \min[2, 0.75 \cdot (2 + 1)] \right\rfloor \\&= \left\lfloor \max[1, 1.5] - \min[2, 2.25] \right\rfloor \\&= \left\lfloor 1.5 - 2 \right\rfloor \\&= \left\lfloor -0.5 \right\rfloor \\&= 0.5\end{aligned}$$

With the above we can calculate the weighted value of each pixel in f_{in} contributing to the output pixel.

for $i' = 0$ and $j' = 1$,

$$\begin{aligned}w_x(0,0) \cdot w_y(1,1) \cdot f_{in}(0,1) &= 0.75 \cdot 0.5 \cdot 10 \\&= 3.75\end{aligned}$$

for $i' = 0$ and $j' = 2$,

$$\begin{aligned}w_x(0,0) \cdot w_y(1,2) \cdot f_{in}(0,2) &= 0.75 \cdot 0.5 \cdot 10 \\&= 3.75\end{aligned}$$

for $i' = 1$ and $j' = 1$,

$$\begin{aligned}w_x(0,1) \cdot w_y(1,1) \cdot f_{in}(1,1) &= 0.25 \cdot 0.5 \cdot 100 \\&= 12.5\end{aligned}$$

for $i' = 1$ and $j' = 2$,

$$\begin{aligned}w_x(0,1) \cdot w_y(1,2) \cdot f_{in}(1,2) &= 0.25 \cdot 0.5 \cdot 100 \\&= 12.5\end{aligned}$$

By summing the above results we get the final value of the output pixel:

$$3.75 + 3.75 + 12.5 + 12.5 = 32.5$$

The final value is rounded to the nearest integer for a value of 33. The remaining pixels in the output image can be found in the same fashion. The resulting 3 x 3 is identical to that of Figure 4 in example 1.

As mentioned before, to complete the degradation process, the lower resolution image must be mapped (transformed) back to the size of the original image. The transform equation above can be applied to the smaller image and used to map it back to the original image size. In the case of the simple example we have been using the 3 x 3 becomes f_{in} and a new 4 x 4 is f_{out} . Therefore, $I = J = 4$ and $I' = J' = 3$. By applying the transform equation to the 3 x 3 the resulting 4 x 4 is identical to Figure 7.

Appendix A shows an algorithm developed to implement the area-weighted average transform. Inputs to the algorithm are: a two-dimensional image of discrete values, the number of rows and columns in the input image, and the number of rows and columns in the desired reduced resolution image. Output from the algorithm is a modified two-dimensional image of the same size as the input image. The procedure which implements the area-weighted average transform equation is executed twice. The first pass transforms the image to reduced resolution size. The second pass reverses the process and returns the image to the size of the input image.

Example 3.

This example shows the input, intermediate result, and output of the algorithm in Appendix A. Figure 9 shows a two-dimensional array of 16 rows and 16 columns which represents an input image. The objective is to degrade this to an apparent resolution of 11 x 11. Figure 10 shows the result of the first pass through the algorithm. This intermediate result is a two-dimensional image of 11 rows and 11 columns. The output of the algorithm is shown in Figure 11. Notice how the degradation process tends to smooth or smear the edges of the square in the image.

10	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10
10	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10
10	10	100	100	100	100	100	100	100	100	100	100	100	100	10	10
10	10	100	100	100	100	100	100	100	100	100	100	100	100	10	10
10	10	100	100	10	10	10	10	10	10	10	10	10	100	100	10
10	10	100	100	10	10	10	10	10	10	10	10	10	100	100	10
10	10	100	100	10	10	10	10	10	10	10	10	10	100	100	10
10	10	100	100	10	10	10	10	10	10	10	10	10	100	100	10
10	10	100	100	10	10	10	10	10	10	10	10	10	100	100	10
10	10	100	100	10	10	10	10	10	10	10	10	10	100	100	10
10	10	100	100	10	10	10	10	10	10	10	10	10	100	100	10
10	10	100	100	10	10	10	10	10	10	10	10	10	100	100	10
10	10	100	100	100	100	100	100	100	100	100	100	100	100	10	10
10	10	100	100	100	100	100	100	100	100	100	100	100	100	10	10
10	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10
10	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10

Figure 9. 16 x 16 Input image.

10	10	10	10	10	10	10	10	10	10	10
10	45	66	66	66	66	66	66	66	45	10
10	66	94	78	78	78	78	78	94	66	10
10	66	78	10	10	10	10	10	78	66	10
10	66	78	10	10	10	10	10	78	66	10
10	66	78	10	10	10	10	10	78	66	10
10	66	78	10	10	10	10	10	78	66	10
10	66	78	10	10	10	10	10	78	66	10
10	66	94	78	78	78	78	78	94	66	10
10	45	66	66	66	66	66	66	66	45	10
10	10	10	10	10	10	10	10	10	10	10

Figure 10. 11 x 11 Intermediate result.

10	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10
10	20	30	41	41	41	41	41	41	41	41	41	41	30	20	10
10	30	49	69	68	67	67	67	67	67	67	68	69	49	30	10
10	41	69	94	84	78	78	78	78	78	78	84	94	69	41	10
10	41	68	84	53	35	35	35	35	35	35	53	84	68	41	10
10	41	67	78	35	10	10	10	10	10	10	35	78	67	41	10
10	41	67	78	35	10	10	10	10	10	10	35	78	67	41	10
10	41	67	78	35	10	10	10	10	10	10	35	78	67	41	10
10	41	67	78	35	10	10	10	10	10	10	35	78	67	41	10
10	41	67	78	35	10	10	10	10	10	10	35	78	67	41	10
10	41	67	78	35	10	10	10	10	10	10	35	78	67	41	10
10	41	68	84	53	35	35	35	35	35	35	53	84	68	41	10
10	41	69	94	84	78	78	78	78	78	78	84	94	69	41	10
10	30	49	69	68	67	67	67	67	67	67	68	69	49	30	10
10	20	30	41	41	41	41	41	41	41	41	41	30	20	10	
10	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10

Figure 11. Resulting 16 x 16 image.

V. APPLICATION OF A SPATIAL TRANSFORM TECHNIQUE

A simpler and more efficient method to achieve the same results as the area-weighted averaging technique is described in the paper by Fant [2]. Fant's technique is based on a two-pass spatial transform technique and has more applications than the scaling required here. It can be used for rotation, true-perspective mapping, mapping to an arbitrary surface, and other applications.

Previous spatial and coordinate transform techniques such as those presented by Schowengerdt [7] and Castleman [1] concentrated on mapping points in the output image to points in the input image. Interpolating the pixel intensities about the mapped point to determine the output pixel value was of secondary importance. Fant's technique reverses the order of importance,

by considering first the intensity interpolation technique and second the way to impose spatial significance on the interpolation technique.²

Fant's technique was considered important for this work for two reasons. First, it is a resampling technique and as was mentioned earlier, degrading an image to lower resolution requires resampling. Second, Fant's technique permits scaling to any arbitrary degree; the scaling factor is not restricted to an integer value. This is important to the AFHRL studies since it permits greater flexibility in the level of degradation.

Fant's two-pass technique is based on a one-dimensional resampling interpolation algorithm. The algorithm maps a line of input pixels onto a line of output pixels. The mapping depends on a sizing factor of the output line in relation to the input line. Figure 12 is a restatement of Fant's one-dimensional resampling interpolation algorithm. It is a more structured and code-oriented than what is presented by Fant.

²Fant [2], page 1.

Variables used in this algorithm are:

SIZEFAC - the size factor
INSFAC - the inverse size factor. How much of an
input pixel contributes to an output
pixel.
INSEG - how much of the current input pixel is
available.
OUTSEG - how much of the current input pixel is
needed to complete an output pixel.
ACCUM - used to build the output pixel value
INVAL - the value of the current input pixel
OUTVAL - the value of the output pixel

Initial values are:

ACCUM = 0
INSEG = 1.0
OUTSEG = INSFAC
INVAL = First input pixel value

Begin

Repeat until all input pixels used and
all output pixels produced.

If OUTSEG <= INSEG Then Do {Produce Output Pixel}
ACCUM = ACCUM + (INVAL * OUTSEG)
INSEG = INSEG - OUTSEG
OUTSEG = INSFAC
OUTVAL = ACCUM * SIZEFAC
ACCUM = 0
End If

Else Do {Use up Input Pixel}
ACCUM = ACCUM + (INVAL * INSEG)
OUTSEG = OUTSEG - INSEG
INSEG = 1.0
INVAL = Next input pixel
End Else

End Repeat

End

Figure 12. One-dimensional Resampling Interpolation Algorithm

Example 4.

This example is similar to the first example presented in Fant's paper. In this example, a line of four inputs pixels is shrunk into a line of three output pixels. This means a size factor of 0.75 and an inverse size factor of 1.33. Figure 13 shows the input line, output line, initial values, and intermediate values through each iteration of the algorithm.

100	110	120	130
-----	-----	-----	-----

Input

102	115	127
-----	-----	-----

Output

Initial Values:

SIZFAC = 0.75 INSEG = 1.0 ACCUM = 0

INSFAC = 1.33 OUTSEG = 1.33 INVAL = 100

		INVAL	INSEG	OUTSEG	ACCUM	OUTVAL
Cycle 1	(input)	100	1.00	1.33	0	
			0.00	0.33	100	
Cycle 2	(output)	110	1.00	0.33	100	
			0.67	0.00	136	102
Cycle 3	(input)		0.67	1.33	0	
			0.00	0.66	74	
Cycle 4	(output)	120	1.00	0.66	74	
			0.34	0.00	153	115
Cycle 5	(input)		0.34	1.33	0	
			0.00	0.99	40	
Cycle 6	(output)	130	1.00	0.00	40	
			0.01	0.00	169	127

Figure 13. Example of one-dimensional interpolation.

According to Fant, spatial transformation of a two-dimensional image can be accomplished by performing a series of one-dimensional resampling interpolations in two passes over the image. In the first pass, the interpolation is performed on the vertical columns in the input image to produce an intermediate image. In the second pass, the interpolation is performed on the horizontal rows in the intermediate image to produce the output image.

Fant discusses a four-corner-to-four-corner mapping procedure for using the two-pass two-dimensional spatial transform. This is primarily useful for transforms which involve rotation or translation. Since the resolution degradation application only requires scaling, this procedure can be simplified.

Appendix B shows an algorithm which implements Fant's two-pass spatial transform technique to achieve resolution degradation. It is modified to make it simpler than the one presented by Fant. It has been streamlined to the scaling features required by the degradation process. It does not require the features useful for other types of mappings. Input to the algorithm is a two-dimensional image, the number of rows and columns in the input image, and the number of rows and columns in the desired reduced resolution image. Output from the algorithm is a degraded two-dimensional image of the same size as the input image. The algorithm executes the spatial transform twice. The first time through, the output of the transform is an intermediate image of the dimensions specified in the algorithm input. The second time through the intermediate image becomes the input to the spatial transform. The sizing factor is inverted, thus making the output image the same size as the original input image. Two simple examples will help illustrate the effects of this technique.

Example 5.

In this example, a 4 x 4 image is reduced to an apparent 3 x 3 image. Therefore, the input and output images are 4 x 4 and the intermediate image is a 3 x 3. Figure 14 shows the input image and the resulting intermediate and output images.

Input Image				Intermediate Image			Output Image			
10	10	10	10				16	27	27	16
10	100	100	10	16	33	16	27	61	61	27
10	100	100	10	33	100	32	27	60	60	27
10	10	10	10	16	32	16	16	27	27	16

Figure 14. 4 x 4 pixel image reduced to 3 x 3 pixel image.

Example 6.

In this example a 16 x 16 image is reduced to an apparent resolution of 11 x 11. Figure 15 shows the 16 x 16 input image. Figure 16 shows the 11 x 11 intermediate image. The 16 x 16 output image is shown in Figure 17.

10	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10
10	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10
10	10	100	100	100	100	100	100	100	100	100	100	100	100	10	10
10	10	100	100	100	100	100	100	100	100	100	100	100	100	10	10
10	10	100	100	10	10	10	10	10	10	10	10	10	100	100	10
10	10	100	100	10	10	10	10	10	10	10	10	10	100	100	10
10	10	100	100	10	10	10	10	10	10	10	10	10	100	100	10
10	10	100	100	10	10	10	10	10	10	10	10	10	100	100	10
10	10	100	100	10	10	10	10	10	10	10	10	10	100	100	10
10	10	100	100	10	10	10	10	10	10	10	10	10	100	100	10
10	10	100	100	10	10	10	10	10	10	10	10	10	100	100	10
10	10	100	100	10	10	10	10	10	10	10	10	10	100	100	10
10	10	100	100	100	100	100	100	100	100	100	100	100	100	10	10
10	10	100	100	100	100	100	100	100	100	100	100	100	100	10	10
10	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10
10	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10

Figure 15. 16 x 16 Input image.

10	10	10	10	10	10	10	10	10	10	10
10	45	66	66	66	66	66	66	66	45	10
10	66	95	78	78	78	78	78	95	66	10
10	66	78	10	10	10	10	10	78	66	10
10	66	78	10	10	10	10	10	78	66	10
10	66	78	10	10	10	10	10	78	66	10
10	66	78	10	10	10	10	10	78	66	10
10	66	78	10	10	10	10	10	78	66	10
10	66	95	78	78	78	78	78	95	66	10
10	45	66	66	66	66	66	66	66	45	10
10	10	10	10	10	10	10	10	10	10	10

Figure 16. 11 x 11 Intermediate result.

10	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10
10	20	30	41	41	41	41	41	41	41	41	41	41	30	20	10
10	30	49	69	68	67	67	67	67	67	67	67	68	69	49	30
10	41	69	95	84	78	78	78	78	78	78	78	84	95	69	41
10	41	68	84	53	35	35	35	35	35	35	35	53	84	68	41
10	41	67	78	35	10	10	10	10	10	10	10	35	78	67	41
10	41	67	78	35	10	10	10	10	10	10	10	35	78	67	41
10	41	67	78	35	10	10	10	10	10	10	10	35	78	67	41
10	41	67	78	35	10	10	10	10	10	10	10	35	78	67	41
10	41	67	78	35	10	10	10	10	10	10	10	35	78	67	41
10	41	67	78	35	10	10	10	10	10	10	10	35	78	67	41
10	41	68	84	53	35	35	35	35	35	35	35	53	84	68	41
10	41	69	95	84	78	78	78	78	78	78	78	84	95	69	41
10	30	49	69	68	67	67	67	67	67	67	67	68	69	49	30
10	20	30	41	41	41	41	41	41	41	41	41	41	30	20	10
10	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10

Figure 17. Resulting 16 x 16 image.

VI. COMPARISON OF TECHNIQUES

It is interesting to compare the results in examples 2 and 3 with the results in examples 5 and 6; they are virtually identical. The only disparity appears to be round-off differences. The two techniques, while using different means, achieve the same end.

The chief difference between the two algorithms is speed of execution. Tests of the two techniques on complex images found Fant's technique to be much more efficient. In Fant's technique the computations are relatively simple. Each pixel in the input image is processed exactly twice: once in the vertical pass and once in the horizontal pass. This is independent of the sizing factor. With the area-weighted averaging technique, pixels in the input image may need to be processed several times, depending on the sizing factor. In the simple case presented in example 2, many of the pixels in the input image had to be processed four times because they contributed to four output pixels. The area-weighted averaging technique processes the image in both dimensions at once whereas Fant's technique handles one dimension at a time.

Up until now the effects of these techniques have only been demonstrated on very simple examples. Now the effects will be demonstrated on images similar to those to be used in AFHRL studies. A Silicon Graphics IRIS system which displays 768 x 1024 pixels with 255 gray levels was used to model an F-15 fighter aircraft.

The display software presented a perspective projection of the F-15 model which was made up of approximately 25 polygons. While this is not a highly detailed model, it is representative of imagery to be used in the AFHRL studies. The model display software permits rotation about the x, y, and z-axis as well as translation along the z-axis (i.e. zoom in and zoom out).

To obtain the input test images, the aircraft model is first put into the desired position through rotation and zooming. Then, the pixels within a 200 by 200 pixel section of the screen containing the aircraft are written out, in row-major order, to a binary file with each byte in the file representing one pixel. Figure 18 is a example of one of these images.

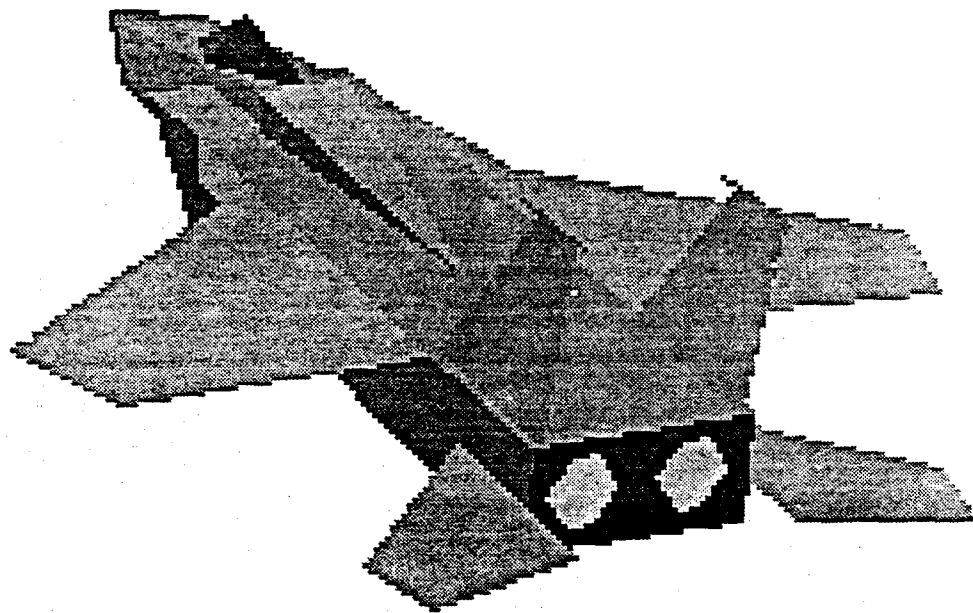


Figure 18. 200 pixel x 200 pixel input image.

Since both resolution degradation techniques yield the same results it is unnecessary to demonstrate both algorithms on actual images. The technique based on Fant's algorithm is used because it is much more efficient.

The image in Figure 18 is the input for the degraded images in Figures 19 through 22. The output images in Figures 19 through 22 are actually 200 pixels by 200 pixels, but the apparent resolution varies. In Figure 19, the input image is degraded to an apparent resolution of 145 pixels by 145 pixels (145 x 145). Therefore, the sizing factor is 0.725. Figure 20 is the input image degraded to an apparent resolution of 110 x 110, for a sizing factor of 0.55. A sizing factor of 0.425 is used to produce the image in Figure 21. Its apparent resolution is 85 x 85. Finally, in Figure 22, the input image is degraded to 60 x 60, for a sizing factor of 0.3.

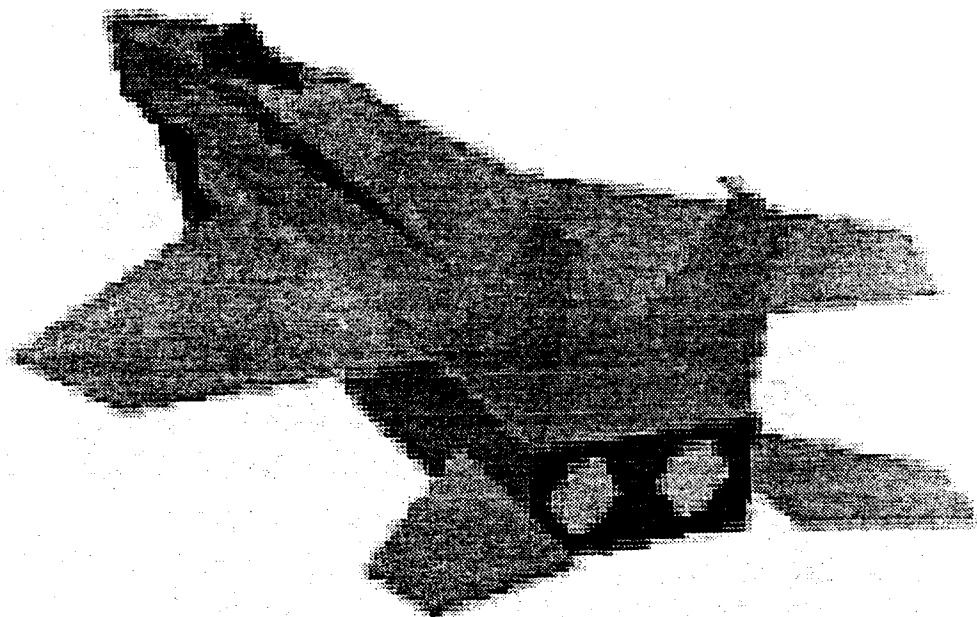


Figure 19. Apparent resolution of 145 x 145.

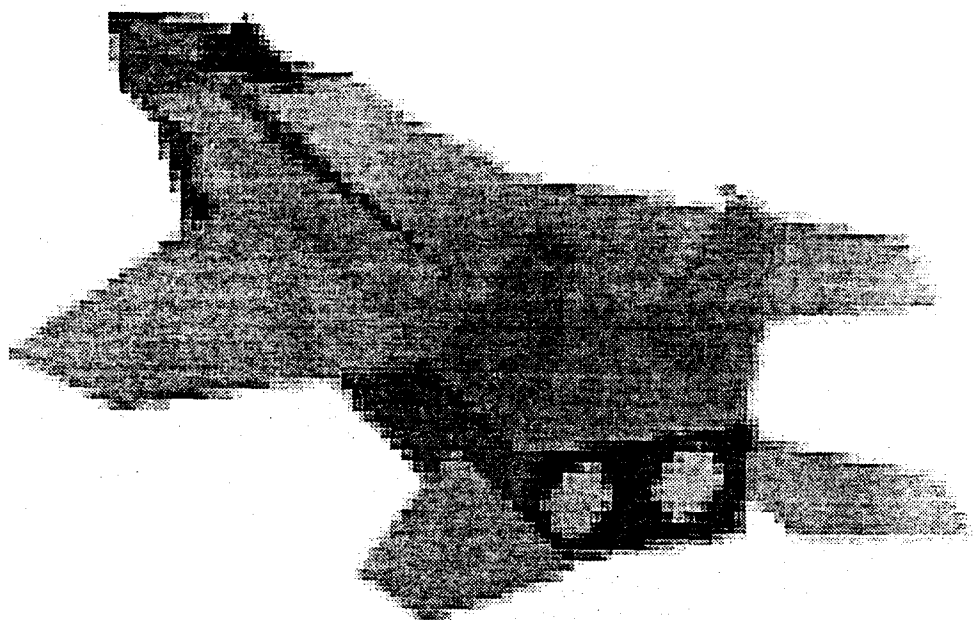


Figure 20. Apparent resolution of 110 x 110.

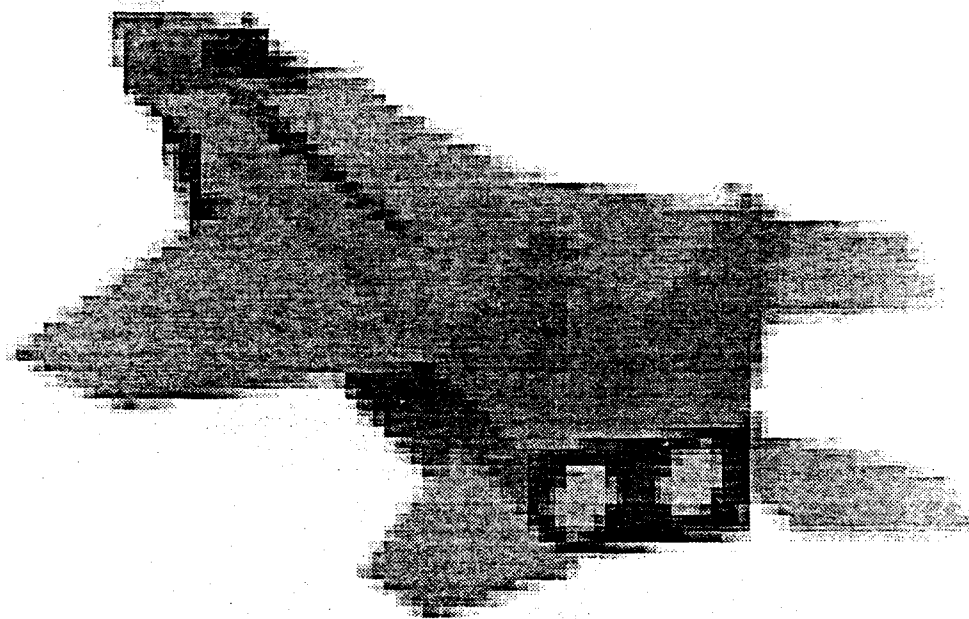


Figure 21. Apparent resolution of 85 x 85.

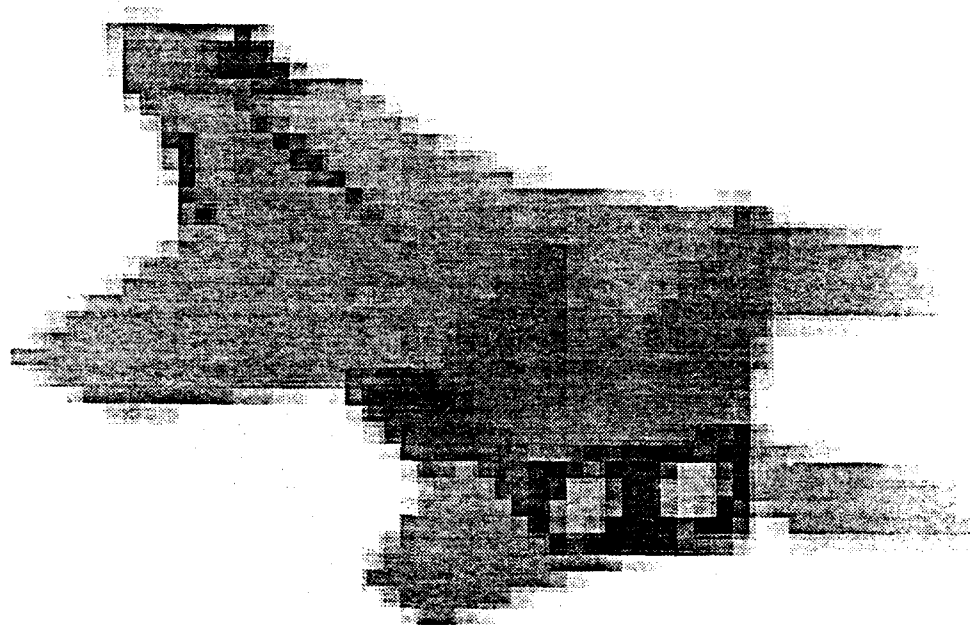


Figure 22. Apparent resolution of 60 x 60.

In looking over the output images some effects can be observed. Edge aliasing (or stair-stepping) is present in the input image and is evident in the output images becoming more and more pronounced as the sizing factor decreases. The resolution degradation presented here is basically a resampling process and aliasing is an expected by-product of resampling at a lower frequency. Therefore, it follows that the lower the resampling frequency, the greater the degree of aliasing.

Another effect which can be observed in the output images is a blurring of the edges. This is especially noticable in edges seperating the aircraft from the background. This effect is present because of the sub-pixel averaging being performed in the degradation algorithms.

Attempts were made to reduce the blurring effects without corrupting other degradation effects. These attempts took the form of post-processing enhancement techniques.

One such technique is edge enhancement using digital convolution filters. Figure 23 shows three filters suggested by Rosenfeld and Kak [5].

These filters were convolved with the output of the resolution degradation process. The unsharp masking filter produced the most satisfactory enhancement. Figures 24 through 27 are images which are the result of applying the unsharp masking filter to the degraded images of Figures 19 through 22. While this process succeeded at enhancing the edges, it typically made the edges much too dark.

Unsharp Masking	High-pass Mask #1	High-pass Mask #2
-1 -1 -1	0 -1 0	1 -2 1
-1 9 -1	-1 5 -1	-2 5 -2
-1 -1 -1	0 -1 0	1 -2 1

Figure 23. Digital Convolution Filters

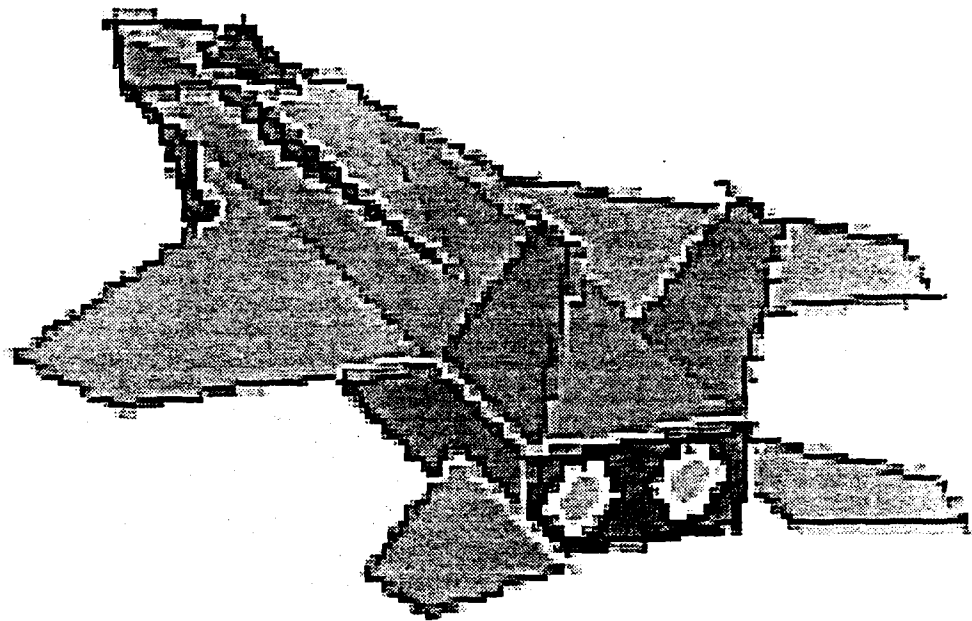


Figure 24. Unsharp masking applied to apparent 145 x 145 image.

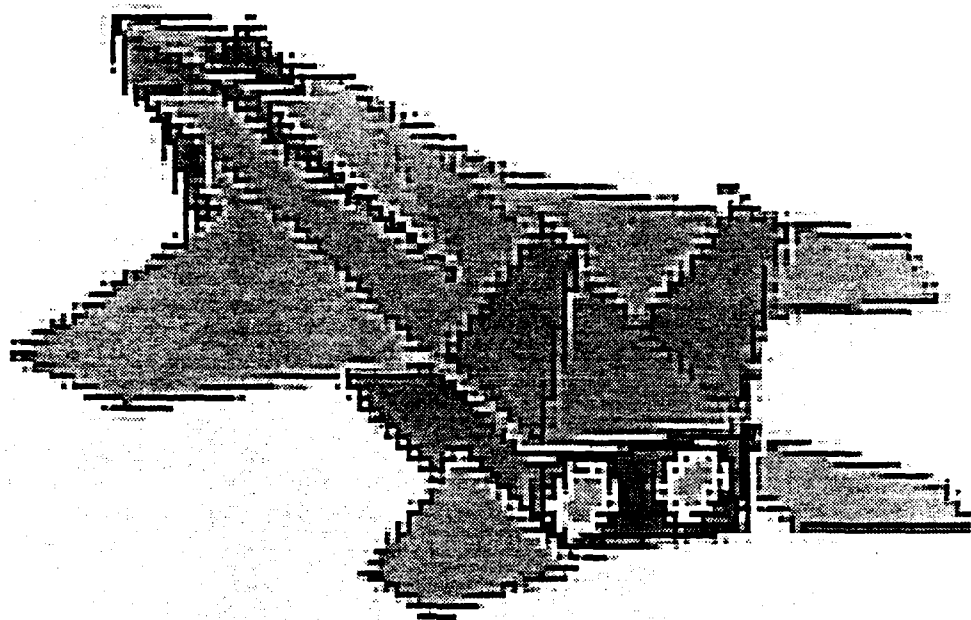


Figure 25. Unsharp masking applied to apparent 110 x 110 image.

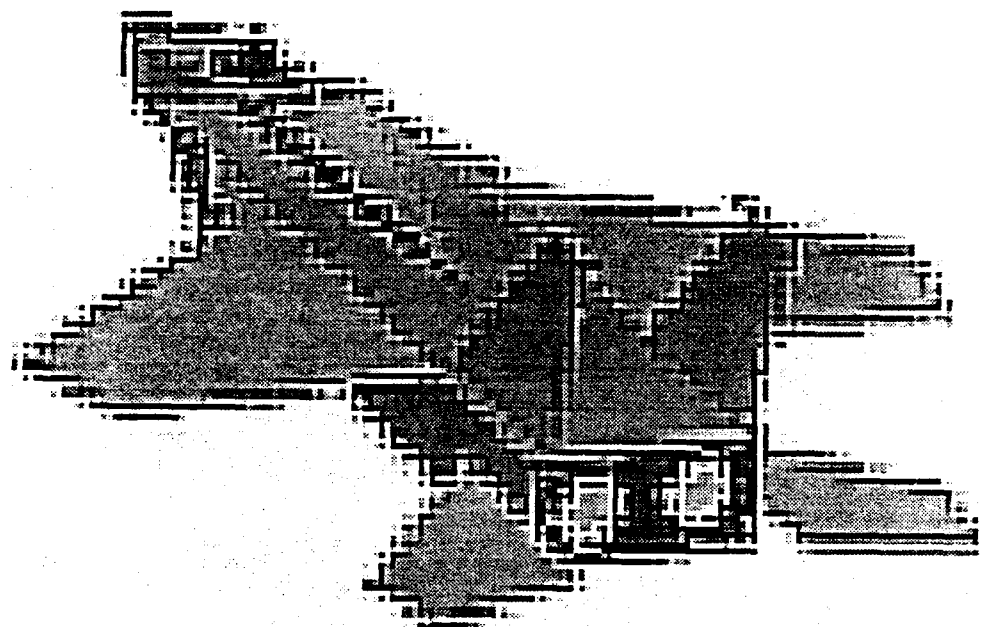


Figure 26. Unsharp masking applied to apparent 85 x 85 image.

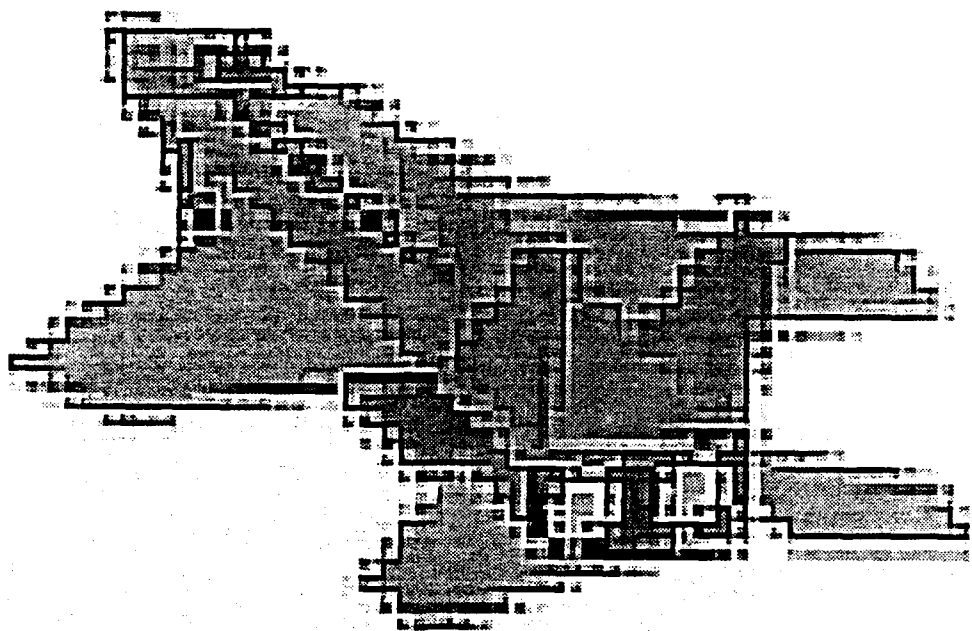


Figure 27. Unsharp masking applied to apparent 60 x 60 image.

Another enhancement technique was found while studying the differences between the input and output images of the degradation process. This technique will be referred to as the "difference enhancement." The following equation describes the difference enhancement:

$$f_E(i, j) = f_{out}(i, j) - |f_{in}(i, j) - f_{out}(i, j)|$$

where $f_E(i, j)$ is the value of the pixel in the i th row and j th column of the enhanced image; $f_{in}(i, j)$ is the value of the corresponding pixel in the input image to the degradation process; and $f_{out}(i, j)$ is the value of the corresponding pixel in the output image of the degradation process.

Figures 28 through 31 show the results of applying the difference enhancement, where the image in Figure 18 is f_{in} and the Figures 19 through 22 are f_{out} , respectively. By comparing the enhanced images with the corresponding output images, one can see increased edge clarity. The difference enhancement does not eliminate the blurriness at edges, but it minimizes it considerably. At the same time it preserves other artifacts of the degradation process. The improvement seems to increase as the *degree of degradation increases*. Since this enhancement is strictly a point operation it could be incorporated directly into the degradation process or left as an optional post-processing step.

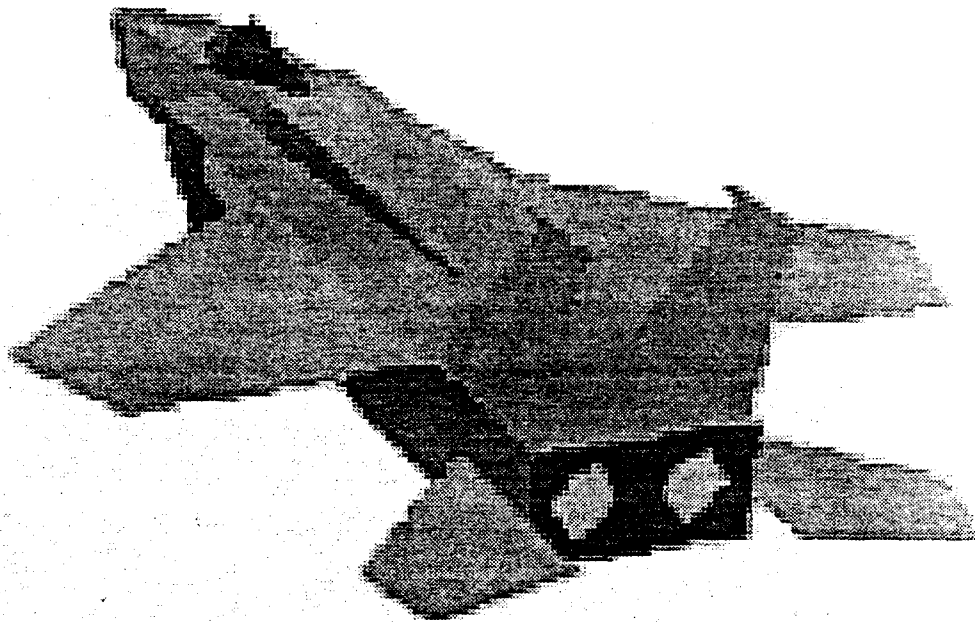


Figure 28. Difference enhancement applied to apparent 145 x 145 image.

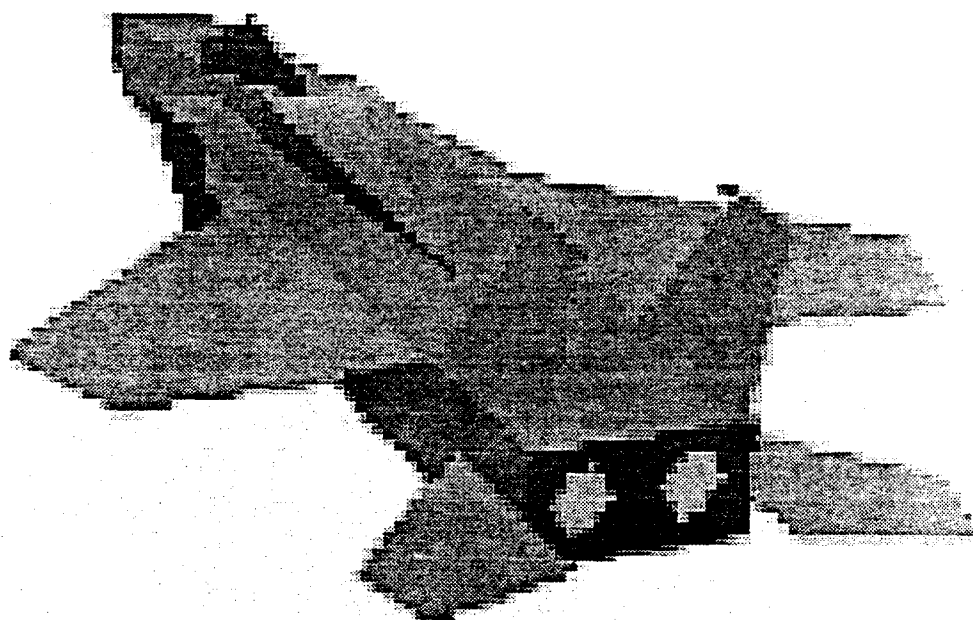


Figure 29. Difference enhancement applied to apparent 110 x 110 image.

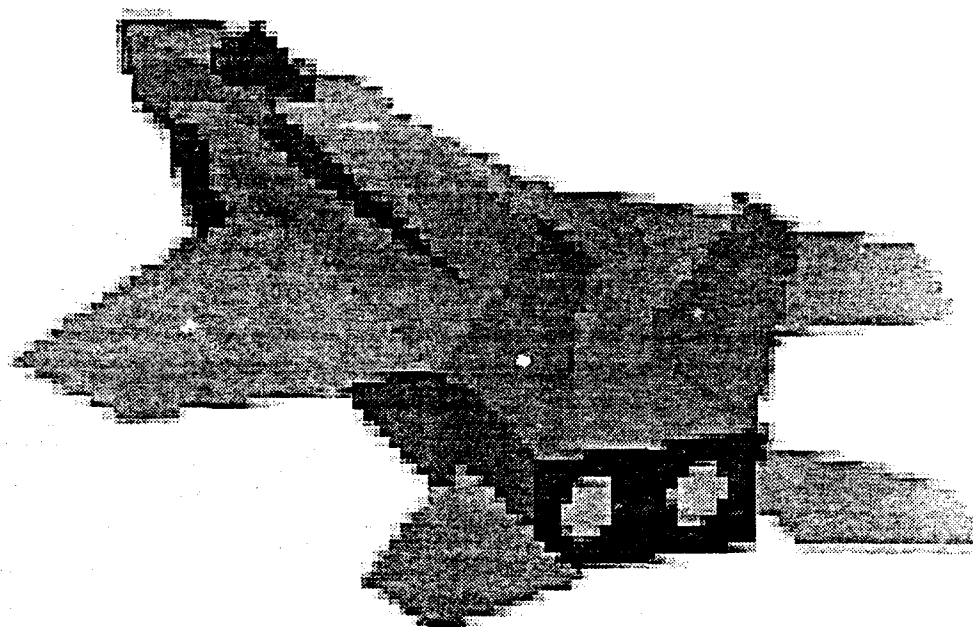


Figure 30. Difference enhancement applied to apparent 85 x 85 image.

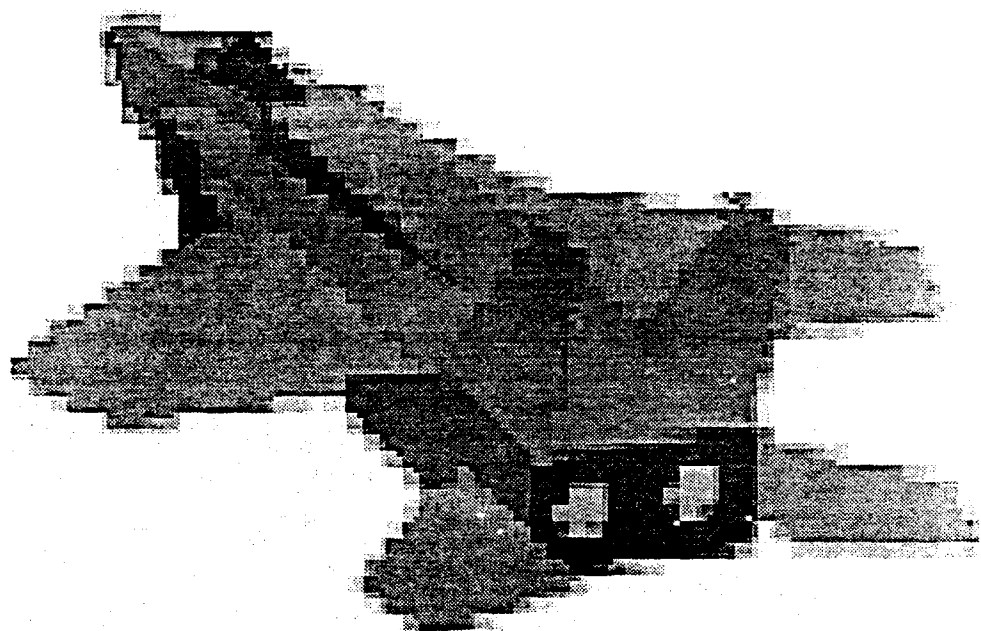


Figure 31. Difference enhancement applied to apparent 60 x 60 image.

VII. AN ALTERNATIVE METHOD

As previously stated, the objective of this work is to take an image displayed on a high resolution screen and modify it to make it appear as though it is displayed on a lower resolution screen. For example, an image displayed on a screen of pixel resolution 768 x 1024 is modified to appear to be displayed on a screen of resolution say 400 x 650. In actuality this 400 x 650 image is being displayed on the 768 x 1024 screen.

The graphics model display software, discussed in the previous section, provides a perspective projection of the F-15 model. The model is displayed through a viewport which has the same dimensions as the display screen (i.e. 768 x 1024). If a display device with a resolution of 400 x 650 could be used in place of the higher resolution display, then the software could display the model using a viewport of 400 x 650. This would be the ideal solution for displaying and studying lower resolution flight simulator images. However, a different display device would be required for each resolution to be studied. This would be impractical. An alternative to the resolution degradation methods presented earlier, however, is based on this idea.

This method, viewport and scale, requires the modification of the software generating the flight simulator images, in this case, the F-15 model. The software must be modified to draw the model in a viewport with dimensions of the desired degraded screen resolution. If the original viewport is not the entire screen, the new viewport should be sized proportional to the desired degree of degradation. It would be preferable to draw into a buffer. The contents of the buffer, an image, become the input to a single pass of either the area-weighted averaging or Fant technique. This single-pass enlarges the image to fit the actual screen display or original viewport. The result is an image of reduced resolution.

To illustrate the advantages of viewport and scale, an image is degraded by viewport and scale and by Fant's two-pass spatial transform technique. Figure 32 is a 300 x 300 image of the F-15 model. Using both methods the image is degraded to an apparent resolution of 90 x 90 (sizing factor of 0.3). Figure 33 shows the results of using Fant's technique for both reduction and enlargement. Figure 34 is the result of using viewport and scale, which uses a smaller viewport to achieve reduction and Fant's

technique to handle enlargement.

Viewport and scale produces a sharper looking image. The image produced by using Fant's technique for both phases is blurred at the edges and the distinction between regions is reduced. In the image produced by viewport and scale the blurriness at the edges is minimal and the regions remain distinct. A significant advantage of viewport and scale is it is faster. The graphics hardware does half of the work resulting in a quick turnaround from input to output.

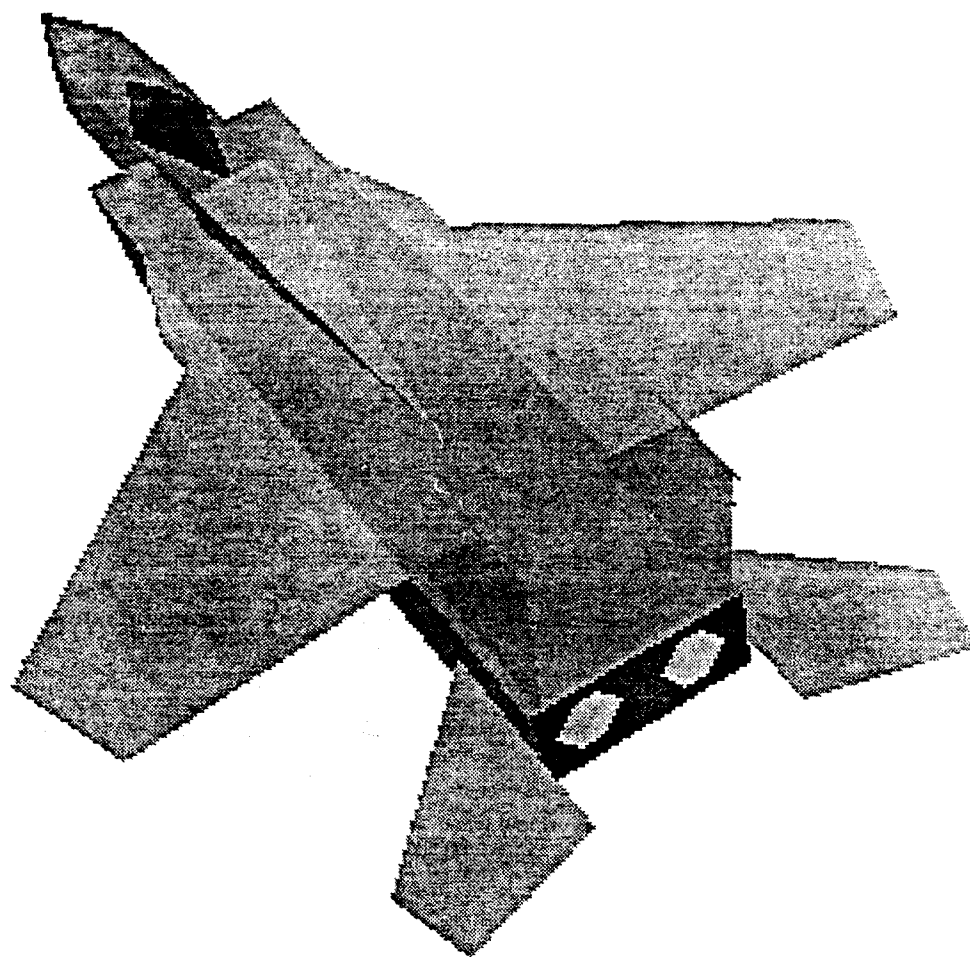


Figure 32. 300 pixel x 300 pixel input image.

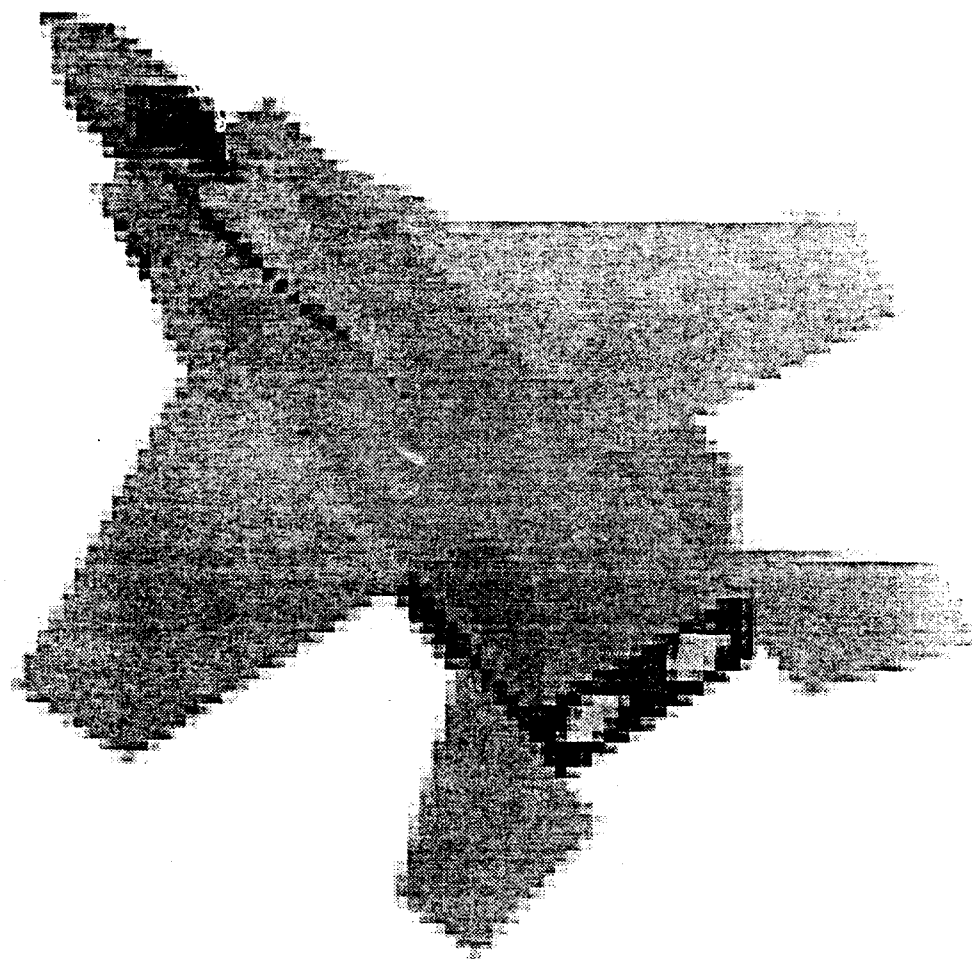


Figure 33. Image degraded to 90 x 90 by Fant's technique.

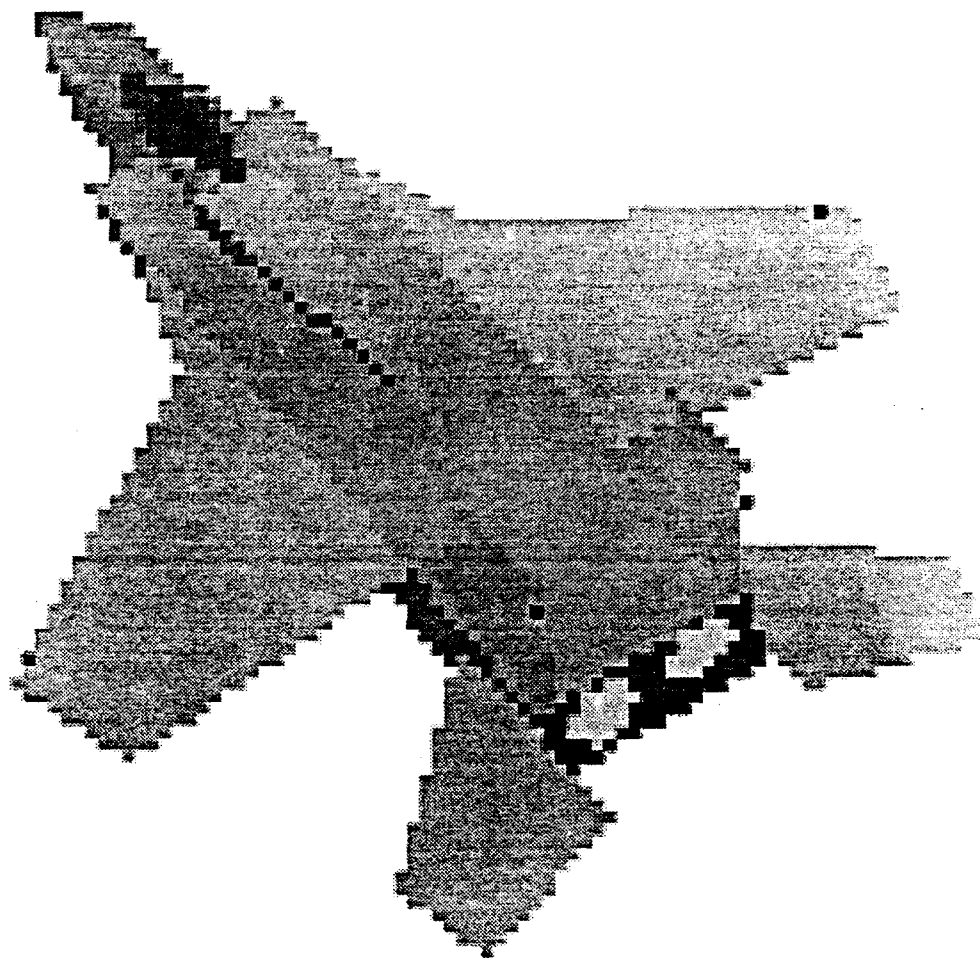


Figure 34. Image degraded to 90 x 90 by viewport and scale.

VIII. SUMMARY

The research presented in this paper was undertaken to provide the Operations Training Division of the Air Force Human Resources Lab a means to degrade the apparent resolution of flight simulator images. The ability to reduce image resolution will support a studies on brightness, contrast, and resolution, which will help to define requirements for future flight simulator visual systems.

Three interrelated techniques were presented to accomplish resolution degradation. All three techniques are based on a two-phased approach. In the first phase the image is resampled, or sized, to fit the dimensions of the reduced resolution image. The second phase takes the image output from the first phase and enlarges or maps it back to the dimensions of the original image. The term "apparent resolution" was used to refer to the resolution of the output image. Actually, the input and output images have the resolution in terms of the number of pixels comprising them. However, the output image "appears" to be made up of fewer and larger pixels.

The area-weighted averaging technique determines the value of output pixels by averaging in the area-weighted contributions of input pixels. The number of pixels and the area weights is dependent on the sizing factor. The second method is based on Fant's two-dimensional spatial transform technique. This consists of two passes of one-dimensional resampling interpolations which depend on the sizing factor.

The first two methods yield identical results. The primary difference between them is the latter is far more efficient. A difference enhancement applied as a post-processing step was found to cure a great deal of blurriness in the edges of the degraded images. These two methods, especially the one based on Fant's spatial transform since it is faster, are the most flexible. They can be applied to any images, whether computer generated or digitized from a photograph.

The third method, viewport and scale, is not so flexible. It is similar to the first two in that the second phase of the degradation process uses one of the first two methods. However, phase one differs because the image generation software must create the image in a smaller viewport. The size of the

viewport is proportional to the degradation sizing factor. Thus, viewport and scale cannot be used on existing digital images. The technique must be incorporated into the image generation software.

The work presented here was on gray-scale images. Most flight simulator systems produce color imagery. Future research in this area should deal with color images. The primary concern would be in handling the pixel averaging where different colors are involved.

LIST OF REFERENCES

1. Castleman, Kenneth R., Digital Image Processing, Prentice Hall, Englewood Cliffs, NJ, 1979.
2. Fant, Karl M., "A Nonaliasing, Real-time Spatial Transform Technique," IEEE Computer Graphics and Applications, January 1986, pp. 71-80.
3. Gonzalez, R. C. and Wintz, P. A., Digital Image Processing, Addison-Wesley, Reading, MA, 1977.
4. Green, William B., Digital Image Processing, A Systems Approach, Van Nostrand Reinhold, New York, 1983.
5. Rosenfeld, A. and Kak, A.C., Digital Picture Processing, Vols. 1 and 2, Academic Press, New York, 1982.
6. Schachter, Bruce J., ed., Computer Image Generation, Wiley Interscience, Wiley Press, New York, 1983.
7. Schowengerdt, Robert A., Techniques for Image Processing and Classification in Remote Sensing, Academic Press, Orlando, Florida, 1983.

APPENDIX A

Algorithm for Resolution Degradation by Area-weighted Averaging Technique

Inputs:

INPUTIMAGE - two-dimensional array of pixel values
INPUTROW - Number of rows of pixels in INPUTIMAGE
INPUTCOL - Number of columns of pixels in INPUTIMAGE
OUTPUTROW - Desired number of rows for degraded image
OUTPUTCOL - Desired number of columns for degraded
image

Outputs:

OUTPUTIMAGE - two-dimensional array of pixel values
representing degraded image

Variable Used:

INTERIMAGE - two-dimensional array of pixel values
representing intermediate image.

Begin:

Call procedure with: { First Pass }

SOURCEIMAGE <- INPUTIMAGE

SOURCEROW <- INPUTROW

SOURCECOL <- INPUTCOL

DESTROW <- OUTPUTROW

DESTCOL <- OUTPUTCOL

DESTIMAGE <- INTERIMAGE

Call procedure with: { Second Pass }

SOURCEIMAGE <- INTERIMAGE

SOURCEROW <- OUTPUTROW

SOURCECOL <- OUTPUTCOL

DESTROW <- INPUTROW

DESTCOL <- INPUTCOL

DESTIMAGE <- OUTPUTIMAGE

End.

Procedure Area-Weighted Average

Inputs:

SOURCEIMAGE - two-dimensional array of pixel values

representing input image to procedure

SOURCEROW - Number of rows of pixels in

SOURCEIMAGE

SOURCECOL - Number of columns of pixels in

SOURCEIMAGE

DESTROW - Desired number of rows of pixels in

image output for procedure

DESTCOL - Desired number of columns in image
output for procedure.

Outputs:

DESTIMAGE - two-dimensional array of pixels
representing output image of procedure

Variables Used:

i - index for rows in DESTIMAGE
j - index for columns in DESTIMAGE
ip - index for rows in SOURCEIMAGE
jp - index for columns in SOURCEIMAGE
imin - minimum index for range of values in
row of SOURCEIMAGE
imax - maximum index for range of values in
row of SOURCEIMAGE
jmin - minimum index for range of values in
column of SOURCEIMAGE
jmax - maximum index for range of values in
column of SOURCEIMAGE
x - vertical weight factor
y - horizontal weight factor
ACCUM - accumulator for summation
INVAL - value of a pixel in SOURCEIMAGE
OUTVAL - value of a pixel in DESTIMAGE
ROWSIZFAC - vertical size factor
COLSIZFAC - horizontal size factor
ROWINSFAC - inverse vertical size factor
COLINSFAC - inverse horizontal size factor

functions used:

`floor(t)` - returns nearest integer less than or
equal to `t`.

`ceil(t)` - returns nearest integer greater than
or equal to `t`.

`max(s,t)` - return the greater of `s` and `t`

`min(s,t)` - returns the lesser of `s` and `t`

`abs(t)` - returns absolute value of `t`

Begin Procedure:

ROWSIZFAC = SOURCEROW / DESTROW

COLSIZFAC = SOURCECOL / DESTCOL

ROWINSFAC = 1 / ROWSIZFAC

COLINSFAC = 1 / COLSIZFAC

for each pixel (OUTVAL) in row i and column j of

DESTIMAGE Do

ACCUM = 0

imin = floor(ROWSIZFAC * i)

imax = ceil(ROWSIZFAC * (i + 1)) - 1

jmin = floor(COLSIZFAC * j)

jmax = ceil(COLSIZFAC * (j + 1)) - 1

for each pixel in row ip of SOURCEIMAGE from imin to

imax and column jp of SOURCEIMAGE from jmin to jmax Do

INVAL = Value of pixel in row ip and column jp of

SOURCEIMAGE

x = abs(max(i, ROWINSFAC * ip)

- min(i + 1, ROWINSFAC * (ip + 1)))

y = abs(max(j, COLINSFAC * jp)

- min(j + 1, COLINSFAC * (jp + 1)))

ACCUM = ACCUM + (INVAL * x * y)

end for

OUTVAL = ACCUM

end for

end procedure

APPENDIX B

Algorithm Based on Fant's Two-pass

Spatial Transform Technique

Inputs:

INIMAGE - two-dimensional array representing the input
image

IROW - Number of rows in input image

ICOL - Number of columns in input image

OROW - Number of rows desired in output image

OCOL - Number of columns desired in output image

Outputs:

OUTIMAGE - two-dimensional array representing the
output image

Other variables:

SIZFAC - the size factor

INSFAC - the inverse size factor. How much of an
input pixel contributes to an output pixel

INSEG - How much of current input pixel is available

OUTSEG - How much of current input pixel is needed
to complete an output pixel

INVAL - value of a pixel in INIMAGE

OUTVAL - value of a pixel in OUTIMAGE

Begin:

$SIZFAC = OROW / IROW$

$INSFAC = 1 / SIZFAC$

{ First pass - vertical }

For each column in input image Do

$INVAL$ = first pixel in input image column

$OUTVAL$ = first pixel in intermediate image column

$ACCUM = 0$

$INSEG = 1.0$

$OUTSEG = INSFAC$

 Repeat until all input pixels in current column used

 and all output pixels in current column of intermediate image produced.

 If $OUTSEG \leq INSEG$ Then Do { Produce output pixel }

$ACCUM = ACCUM + (INVAL * OUTSEG)$

$INSEG = INSEG - OUTSEG$

$OUTSEG = INSFAC$

$OUTVAL = ACCUM * SIZFAC$

$ACCUM = 0$

$OUTVAL$ = Next output pixel in intermediate image

 End If

 Else Do { Use up input pixel }

$ACCUM = ACCUM + (INVAL * INSEG)$

$OUTSEG = OUTSEG - INSEG$

$INSEG = 1.0$

$INVAL$ = Next input pixel

 End Else


```

End Repeat

End For

{ Intermediate Image Produced }

SIZFAC = OCOL / ICOL

INSFAC = 1 / SIZFAC

{ Second pass - horizontal }

For each row in intermediate image Do

    INVAL = first pixel in intermediate image row

    OUTVAL = first pixel in output image row

    ACCUM = 0

    INSEG = 1.0

    OUTSEG = INSFAC

    Repeat until all input pixels in current row used and
    all output pixels in current row produced

        If OUTSEG <= INSEG Then Do { Produce output pixel }

            ACCUM = ACCUM + (INVAL * OUTSEG)

            INSEG = INSEG - OUTSEG

            OUTSEG = INSFAC

            OUTVAL = ACCUM * SIZFAC

            ACCUM = 0

            OUTVAL = Next output pixel

        End If

    Else Do { Use up input pixel }

        ACCUM = ACCUM + (INVAL * OUTSEG)

        OUTSEG = OUTSEG - INSEG

        INSEG = 1.0

```

INVAL = Next input pixel

End Else

End Repeat

End For

{ Output Image Produced }

End